Catch Your Breath: Adaptive Computation for Self-Paced Sequence Production

Alexandre Galashov[*]
Google DeepMind

Matt Jones
Google DeepMind

Rosemary Ke
Google DeepMind

Yuan Cao
Google DeepMind

Vaishnavh Nagarajan
Google DeepMind

Michael C. Mozer
Google DeepMind

Abstract

Within the landscape of inference-time scaling methods for foundation models, a *width-based* approach to scaling—which involves the insertion of <PAUSE> tokens in the input stream to delay model responses—offers a unique advantage by increasing model expressivity while remaining highly parallelizable at both training and inference. The existing literature on training models to utilize <PAUSE> tokens relies on the standard cross-entropy objective in which the model output is read out and evaluated only at the final step of a pause sequence. This approach provides no mechanism for the model to regulate its own processing or to signal readiness to respond, treating the additional compute steps as a barrier to skip over rather than a resource to be used adaptively. We propose a supervised loss, *Catch Your Breath* (*CYB*), framed as a sequential-decision problem, that trains a model to dynamically and autonomously scale the number of compute steps used for each input token. The model indicates the need for additional compute steps by emitting a special <DON'T KNOW> output, delaying its response via a pause. The model can abstain multiple times to obtain longer delays. Our experiments demonstrate that when <PAUSE> tokens are included in the input stream, pretraining or fine-tuning with CYB significantly outperforms standard cross-entropy, reducing perplexity and enhancing downstream accuracy with no additional computational or memory cost.

While remarkable advances in AI have been achieved by scaling model size and dataset size, a third scaling dimension has recently become a focus of intense interest: scaling the computational budget during inference. The most straightforward approach to inference-time scaling involves 'thinking' in some form, whether chain of thought [e.g., 45, 48, 50], repeated sampling, self critiques, or latent-thought methods [e.g., 16, 37, 40].

Inference-time scaling can also operate at a more granular level by boosting resources devoted to individual tokens, either by expanding model *depth* or model *width*. Depth-based approaches involve layer recurrence, i.e., repeating individual layers or a range of layers [e.g., 13]. Width-based approaches involve inserting special <PAUSE> tokens into the input stream [14]. Each <PAUSE> informs the model that it has an additional forward pass to perform further processing on the context and potentially improve the next token emitted.

Although depth-based approaches have gotten the bulk of notice in the literature (see Section 1), our work focuses on width-based approaches which offer a complementary form of expressivity [29, 31] and can be parallelized, unlike depth-based approaches. However, width-based approaches come with a critical drawback which our work aims to address. In width-based approaches, processing time is expanded for an input token by inserting one or more trailing <PAUSE> tokens, and when these tokens

[*]Address correspondence to mcmozer@google.com

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

occur, the model output is ignored until the final pause step [14]. To appreciate the computational challenge this insertion poses for a transformer, consider an indexed token sequence such as

`0:Donald, 1:lost, 2:his, 3:<pause>, 4:<pause>, 5:mind`.

Had there been no pauses here, the model would have been trained at step 2 to predict `mind`; with the pauses at steps 3 and 4, the model is trained to output `mind` at step 4 instead. However, this causes two troubles at inference time. First, even if the model already predicts well at step 2, it must hold its response for two additional steps. Additional computation time is not always helpful. Second, because the model does not control the number of pauses, it may act greedily and try to predict at each step, which may deny it the opportunity to plan optimally with the additional pauses in cases where more compute is beneficial.

To remedy these limitations of externally imposed delays, we propose a method enabling the model to autonomously insert variable-duration delays as needed. This method is inspired by studies of human reading, which find that readers fixate longer on words when processing demands rise (see Appendix A). For each input token, the model is given the option of emitting a special <DON'T KNOW> token to abstain from responding. With abstention, the arrival of further input is held off and a <PAUSE> token in the stream signals continued processing (Figure 1a). The model can produce abstention responses multiple times in a row, yielding adaptive self-paced sequence production (Figure 1b).

To summarize our contributions: (1) We propose a principled loss, called *Catch Your Breath* or *CYB*, that trains a transformer to self-determine how many computation steps it requires for each input token. While most adaptive-computation techniques focus on inference acceleration, we show that training pause tokens with CYB yields uniformly superior performance compared to the existing (non-adaptive) method in the literature—applying the standard cross-entropy objective on the last pause step. (2) We propose a parallel training procedure for the CYB loss that is as compute and memory efficient as cross entropy. (3) We explore variants of the loss that include additional penalties for slow responses but find—consistent with our conjecture—that the CYB loss naturally drives the model to respond efficiently. (4) We show significant advantages of the CYB loss over cross entropy, both for models pretrained from scratch and models fine-tuned with <PAUSE> tokens, on both perplexity measures and downstream model performance. (5) We investigate where the model chooses to pause, and find systematic and intuitive insights into the model's behavior.

1 Related research

Pause tokens. Prior work has proposed special content-free input tokens to improve model performance, variously referred to as 'memory' [6], 'thinking' [19], 'dummy' [25, Appendix A.3], 'registers' [10], and 'filler' [36] tokens. Goyal et al. [14] coined the term 'pause' tokens and focused on their role in granting models additional computation steps. They were the first to conduct extensive experiments with pretraining and fine-tuning, and subsequent work has utilized their paradigm. We will refer to their approach by the title of their paper, *Think Before You Speak* or *TBYS*. During pretraining, their data loader inserts M pauses into the text stream at uniformly random positions. When pauses are inserted, model outputs for the $M-1$ steps leading up to the final pause token are ignored and excluded from the usual next-token-prediction cross-entropy loss. During fine-tuning, where a prefix sequence is given along with a target sequence, M' pauses are appended to the prefix, giving the model additional compute steps before its response. Goyal et al. found consistent benefits in downstream-task accuracy only when pauses were included in both pretraining and fine-tuning stages. Although this

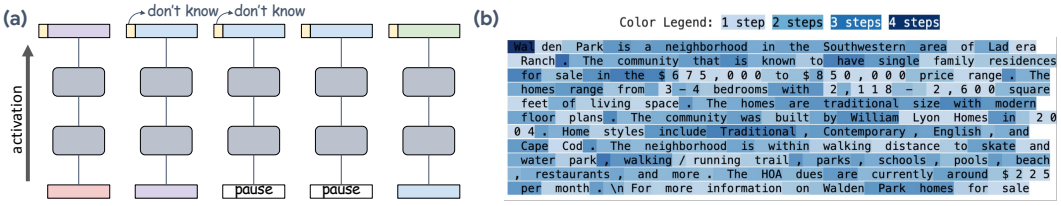

Figure 1: (a) Sequence processing in a transformer with <DON'T KNOW> outputs that trigger <PAUSE> inputs. (b) The CYB loss yields variable per-token computing steps. When the <DON'T KNOW> probability is high, additional computing steps are provided. Shading indicates the expected number of computing steps to predict the corresponding token. Spacing reflects token boundaries.

approach appears to be an alternative to chain-of-thought methods, pauses may be related given that substituting meaningless filler tokens for thinking still yields benefits [36]. In previous work, long delays are introduced between a prefix and a model's response, intended to support explicit reasoning about and calculation of the response—an alternative to chain of thought. In contrast, our motivation is to allow short delays in processing any token, with the aim of facilitating integration of information in prefix tokens as well as in choosing any token in the response. Similar in spirit to our proposal, Dong et al. [11] propose an RL method that generates reasoning traces for individual tokens, also aimed at next-token prediction. The computational cost of our technique is significantly lower.

Anytime prediction and adaptive computation. In anytime prediction, models make a series of responses that progressively improve with additional compute. Such models exhibit a *speed-accuracy* trade-off in which a coarse answer appears quickly but is refined with additional computation. In some schemes, the model is rewarded for getting the right response in as few steps as possible [e.g., 20], similar to adaptive computation time (ACT) for recurrent networks [15]. However, whereas ACT and anytime prediction focus on dynamically terminating inference to reduce wall-clock latency, our approach uses a sequential-decision framework to adapt computation within a fixed parallel-step budget. PonderNet [4] applies probabilistic halting over serial depth recurrence and minimizes expected loss regularized by a geometric KL prior, whereas CYB operates over parallel input steps and minimizes log of expected discounted accuracy without a KL penalty (though see Appendix C).

Adaptive depth models. Extending adaptive computation-time to transformers, *looped transformers* and variants have been proposed in which the functional depth of a model can be varied by repeating individual layers or ranges of layers [1, 3, 7, 12, 21, 24, 27, 30, 33, 34, 39, 47, 49, 53, 54]. Adaptation can be made on a per-token basis with a learned router. Looping increases model depth, whereas pause tokens increase model width (input steps). Because depth increases require serial computation, looping cannot exploit hardware parallelism like pause tokens can. Looping and pausing extend model expressivity in complementary ways [29, 31], and thus should be considered independent possibilities rather than competing alternatives.

<DON'T KNOW> outputs. The <DON'T KNOW> output is related to classification with abstention (or rejection) [5, 17, 28]. Two recent articles introduce heuristic notions of "I don't know" to determine where pause tokens should be inserted. Cohen et al. [9] include a <DON'T KNOW> output and introduce a heuristic training objective that shifts probability to the <DON'T KNOW> output when the model predicts incorrectly. Their approach applies multiple heuristics to prevent the degenerate solution of always outputting <DON'T KNOW>. By integrating the <DON'T KNOW> option into our sequential-decision task, our approach does not suffer such degenerate solutions. Kim et al. [23] propose a heuristic method, DIT, for dynamically inserting pause tokens based on low token likelihood (confidence). Their training procedure relies on this proxy as the basis for inserting pauses, whereas CYB learns where additional compute actually improves accuracy via a principled sequential-decision loss. DIT is based on a two-pass training procedure; in contrast, CYB requires only a single pass, albeit with a larger number of <PAUSE> tokens. Finally, DIT cannot distinguish aleatoric from epistemic uncertainty, which the CYB loss can in principle infer. Because DIT and CYB differ on so many dimensions, a compute-and-memory-matched comparison is highly challenging.

2 Optimizing compute steps with the Catch-Your-Breath (CYB) loss

We propose a loss that lets a transformer *autonomously* determine the number of compute steps used for individual tokens. (If notation in this section is challenging, Table B.1 provides a glossary.)

In prior work with pause tokens, each input token is allotted up to M compute steps by inserting between 0 and $M-1$ <PAUSE> tokens into the input stream. Specifically, an external controller, which we refer to as the *world*, draws the allotted number of steps $i \in \{1, 2, \ldots, M\}$ from a prior distribution $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_M)$, where ω_i is the probability that a token is granted exactly i steps ($i-1$ pauses) to respond. A training loss is applied at the final step. Under causal masking, however, the model cannot anticipate how many pauses follow the current step. It must therefore be prepared to respond at each step, leading to greedy optimization that may not best leverage the allocated compute time. Following [14], we refer to this state-of-the-art baseline approach as *Think Before You Speak* (*TBYS*).

We hypothesize that pause steps might be better utilized if the model itself can control whether pauses are used. To this end, our approach, CYB, lets the model autonomously expand the processing of

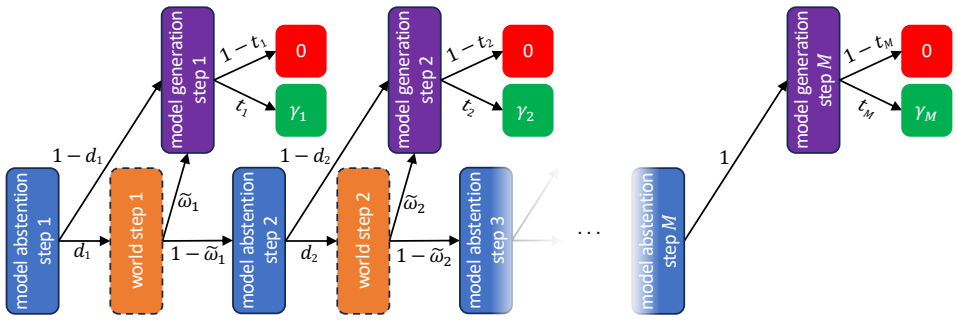

Figure 2: Catch Your Breath: A sequential-decision framework to obtain a single output token. The blue boxes represent model abstention decisions, the purple boxes model choices of output tokens. The orange boxes are choices made by the environment about whether the model is granted a delay in responding. The green and red boxes represent end-state rewards. The unconditional stop-probability at step i, ω_i, has been rewritten as a hazard probability $\tilde{\omega}_i = \omega_i / \sum_{j=i}^{M} \omega_j$.

each token up to as many as M steps. However, the model can stop sooner: at each step $i < M$, the model can choose either to respond or to abstain via a <DON'T KNOW> response. If the model chooses to respond or $i = M$ (the deadline is reached), a token is drawn from the usual next-token distribution, excluding <DON'T KNOW>.

We frame the selection of each generated token as a sequential-decision problem, depicted in Figure 2. The subscripts in the Figure are a step index relative to the presentation of the real input token, $i \in \{1, 2, \ldots, M\}$. The model has a sequence of decision points, indicated by the blue and purple boxes. At each step i, the model selects <DON'T KNOW> with probability $d_i \in [0, 1]$, a value that is dependent on model parameters and input. In accordance with the model's confidence in its answer (probability $1 - d_i$), it selects an output token. The selected token will be either the target, with probability t_i, or an incorrect response, with probability $1 - t_i$. The green squares indicate the reward (accuracy) when correct, discounted by $\gamma_i \in [0, 1]$ at step i, and the red squares are the non-reward when incorrect.

Our sequential-decision framework allows for the external world to terminate processing early (as in TBYS). It incorporates components, depicted by the dashed orange boxes in Figure 2, which permit the world to terminate processing under the prior distribution $\boldsymbol{\omega}$. In most experiments we conduct with CYB, models are allowed to use M steps per token, i.e., $\boldsymbol{\omega} = (0, 0, \ldots, 1)$.

For a given token k, where the model produces <DON'T KNOW> outputs $\boldsymbol{d}^k = (d_1^k, \ldots, d_M^k)$, the step at which the model generates an output—self- or world-induced—is a random variable, S, with probability given by

$$\Pr(S = i \mid \boldsymbol{d}^k, \boldsymbol{\omega}) = \left(\omega_i + (1 - d_i^k) \sum_{j=i+1}^{M} \omega_j\right) \prod_{j=1}^{i-1} d_j^k \tag{1}$$

and $d_M^k \equiv 0$ for the final step. Equation (1) follows from Figure 2; Appendix G has a derivation.

For token k, with model stepwise prediction accuracy $\boldsymbol{t}^k = (t_1^k, \ldots, t_M^k)$, the expected discounted accuracy in the CYB decision task is $\mathbb{E}_{i \sim \Pr(S \mid \boldsymbol{d}^k, \boldsymbol{\omega})} \left[\gamma_i t_i^k\right]$, which can be used to define a generalized negative log likelihood (NLL) loss:

$$\ell_{\text{CYB}}(\boldsymbol{t}^k, \boldsymbol{d}^k;\, \boldsymbol{\omega}, \boldsymbol{\gamma}) = -\log \mathbb{E}_{i \sim \Pr(S \mid \boldsymbol{d}^k, \boldsymbol{\omega})} \left[\gamma_i t_i^k\right]. \tag{2}$$

Averaging over all tokens k, we obtain the standard expectation-of-log NLL formulation:

$$\mathcal{L}_{\text{CYB}} = \mathbb{E}_k \left[\ell_{\text{CYB}}(\boldsymbol{t}^k, \boldsymbol{d}^k;\, \boldsymbol{\omega}, \boldsymbol{\gamma})\right]. \tag{3}$$

This loss is used to optimize model parameters via back propagation through the outputs $\boldsymbol{t}^k$ and the <DON'T KNOW> probabilities $\boldsymbol{d}^k$. It is conditioned on hyperparameter vectors $\boldsymbol{\omega}$, the distribution over the maximum number of steps allotted for a token, and $\boldsymbol{\gamma}$, the step-wise discounting vector.

TBYS [14], the standard cross-entropy loss for <PAUSE> tokens, can be formulated as

$$\ell_{\text{TBYS}}(\boldsymbol{t}^k; \boldsymbol{\omega}) = \mathbb{E}_{i \sim \text{Cat}(\boldsymbol{\omega})} \left[-\log t_i^k \right], \tag{4}$$

where the stop-probability priors, $\boldsymbol{\omega}$, correspond to the manner in which tokens are inserted into the stream for training. For example, if two pauses are randomly inserted following 10% of real tokens, $M = 3$ and $\boldsymbol{\omega} = [.9, 0, .1]$. In our implementation, TBYS is trained with the stop-prior distribution provided to CYB, which simplifies the loss to $\ell_{\text{TBYS}}\left(\boldsymbol{t}^k \mid \boldsymbol{\omega} = (0, 0, \ldots, 1)\right) = -\log t_M^k$.

CYB is motivated by a set of hypotheses, **H1**-**H4**, that we evaluate via simulation experiments.

H1. We conjecture that CYB will outperform TBYS when matched in training data and compute resources because CYB self-determines when to read out whereas TBYS lacks this control.

H2. CYB should yield a well calibrated model: to minimize the loss, the <DON'T KNOW> probability at step i, d_i, should be low when the target probability, t_i, is high and vice-versa. Further, the model should be time efficient: to minimize the CYB loss, the model should respond at step i unless the model's response is likely to improve at step $i + 1$. Thus, it should not request all M steps if not warranted by improved accuracy.

H3. Our hypothesis is that CYB's benefit stems from its control over when to respond, because having this control allows more effective use of computation. Thus, if deadline uncertainty is added via the stop-time distribution, $\boldsymbol{\omega}$, CYB cedes some control (like TBYS), and we predict that accuracy will drop.

H4. The model's speed-accuracy trade-off can be shifted to allow the CYB-trained model to respond in fewer average steps at the expense of accuracy. We can impose a linear cost per step with, say, $\gamma_i = \gamma_0^{i-1}$ for hyperparameter γ_0.

3 Methodology

Position encoding with pause sequences. To encode an input sequence, each token is assigned a position index for use by positional embedding methods such as RoPE [42]. Treating <PAUSE> inputs like any other token will cause them to shift the position indices of following tokens. This approach was used by Goyal et al. [14], but may be problematic when introducing pauses during fine-tuning in a model that was trained without them. Consequently, we use RoPE but assign <PAUSE> tokens the same index as the immediately preceding non-<PAUSE> token. To ensure that the model can selectively index a specific pause token, we encode each pause in the sequence with a unique token code: <PAUSE1>, <PAUSE2>, etc.

Representing abstention. To obtain the <DON'T KNOW> output, d_i, we repurpose an unused token δ from the standard set of output alternatives and then renormalize the model's output distribution, y, as $\hat{y}_{i,j} = y_{i,j}/(1 - y_{i,\delta})$ for token indices $j \neq \delta$, and $\hat{y}_{i,\delta} = 0$ otherwise. Whether fine-tuning or pretraining, we wanted to ensure a large enough <DON'T KNOW> prior probability that the model would learn to use it. We renormalized model posteriors to reflect a high prior for <DON'T KNOW>, as detailed in Appendix D.

Efficient training with CYB. Although the sequential-decision framework (Figure 2) might suggest training the model over sequential steps, we propose a procedure in which $M - 1$ pauses are inserted into the input stream after each real input token during training. The transformer steps are run in parallel with causal masking. The pause steps may or may not be used by CYB, e.g., if at some step i the model is certain ($d_i \to 0$) or the world enforces a stop (via ω_i), the outputs at subsequent pauses will not contribute to the current token's loss. During inference, when the transformer might be run autoregressively, we match the use of pauses at train time, i.e., always inserting $M - 1$ pauses after each real input token. Because an input token and the pauses that follow can be processed in parallel, there is a trivial computational cost to inserting the maximum number of pauses in the stream for every token. Rather, the potential downside comes from the pauses cluttering the context window.

4 Results

We conducted experiments using Gemma1-2B [44] and Gemma3-4B [43], which we fine-tuned on a subset of C4 [38]. We use sequences of 2048 real tokens which we expand to context windows of 4096, 6144, and 8192 tokens by inserting 1, 2, or 3 pauses following each real token, respectively. Unused token indices were repurposed to serve as pauses and <DON'T KNOW>. Unless explicitly

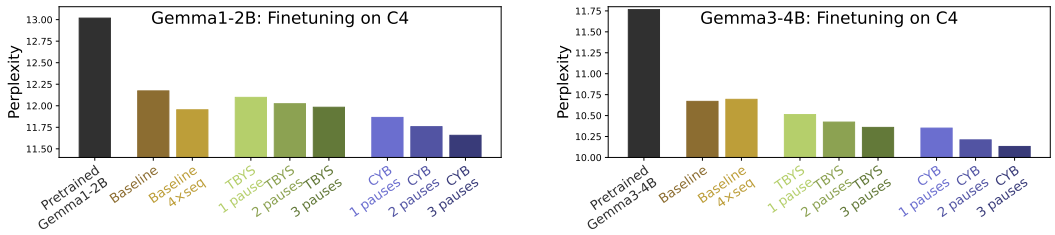

Figure 3: C4 evaluation set perplexity (lower is better) for pretrained Gemma1-2B (**left**) and Gemma3-4B (**right**) models (black bar) and various models fine-tuned on the C4 train set. Model variants include a baseline condition, where no modification is made to training objective or token sequence (brown bar), Think Before You Speak (TBYS) with 1, 2, and 3 pause tokens after each input token (green bars), and the CYB with 1, 2, and 3 pause tokens.

mentioned, all experiments conducted below use fine-tuned Gemma1-2B. Additionally, we conducted a pretraining experiment with Gemma1-2B using C4 [38]. Further details of the training procedure can be found in Appendix D and extended results in Appendix F.

Our experiments focus on comparing two alternative losses for training a model with input <PAUSE> tokens: CYB, our method, and TBYS [14], which uses the standard cross-entropy loss and reads out the model's response at the final <PAUSE>. We match CYB and TBYS in the insertion of pauses, with $n \in \{1, 2, 3\}$ pauses after each real token. The methods differ only in the loss and the token-readout procedure. Compute and memory requirements are identical. Additionally, we train baseline methods on the same data *without the pauses* in order to show that pausing improves model quality. We can either match the baseline and pausing methods (CYB, TBYS) in terms of training data or compute. To match training data, we train a model referred to as *baseline* with a 2048 token context window. To match compute of the three-pause CYB and TBYS models, we train a model referred to as *baseline 4×seq* with an 8192 token context window which is filled with real data, providing it with fourfold advantage in data over the pausing methods.

Fine-tuning experiments. Figure 3 presents perplexity on a C4 evaluation set across different fine-tuned model variants. As one would expect, the pretrained model with no fine-tuning (black bar) has higher perplexity than the baseline model fine-tuned on C4 (brown bar). The light brown bar is a second baseline trained on sequences of length 8192 real tokens, which is compute matched to TBYS and CYB with three pauses. CYB consistently outperforms TBYS, to the point where 1-pause CYB beats 3-pause TBYS. The data-matched baseline (dark brown bar) has higher perplexity than any of the CYB and TBYS methods; the baseline that is compute matched to 3-pause TBYS/CYB (light brown bar) has higher perplexity than CYB with 1, 2, or 3 pauses.

To address reliability concerns, we fine-tuned a baseline Gemma3-4B model to the C4 dataset. We performed three replications of three conditions: baseline (no pauses), CYB with 3 pauses, and TBYS with 3 pauses. We found almost no variance in eval performance across seeds for baseline (10.670, 10.670, 10.669), CYB (10.131, 10.134, 10.129), or TBYS (10.359, 10.360, 10.358). These replications vary only the training data order since the weights come from the pretrained model.

Curiously, Goyal et al. [14] found that TBYS did not improve beyond baseline with fine-tuning. The difference might be explained by several adjustments to their methodology. We assign pause tokens distinct identities depending on their relative positions, and we use the same absolute sequence position for all pauses in a row. Further, we insert pause tokens systematically after every true input token, whereas Goyal et al. inserted them randomly and infrequently.

Pretraining experiment. We also conducted a pretraining experiment on C4, comparing TBYS and CYB with 3 pauses, with baselines included as a reference (Figure 4). As with fine-tuning experiments, we observe superior performance by both TBYS and CYB over the baselines, and CYB delivers the strongest performance. CYB shows a larger benefit over TBYS in the pretraining setting relative to the fine-tuning setting: 3.6% lower perplexity for pretraining versus 2.7% for fine-tuning. (See Tables F.3 and F.4 in Appendix F.1.) Both the fine-tuning and pretraining experiments support our hypothesis **H1** that CYB outperforms TBYS. These methods are matched in memory, data, and computation and differ only in the loss function and the token-readout procedure.

Data efficiency of CYB. While CYB reduces perplexity, developing intuitions about the significance of this reduction is challenging. We aim to offer such intuition by casting the improvement in terms of data efficiency. Figure 6 presents evaluation perplexity of various models fine-tuned in different data regimes. The regimes labeled 1×1 and 2×1 correspond to one and two passes through our 6.4M training sequences, respectively. The regimes labeled 1×2 and 1×3 correspond to data sets that are two and three times as large, respectively. CYB requires just 33% of the data to match baseline performance, and less than 66% of the data to beat performance of TBYS with three pauses. Interestingly, the performance gap between CYB and the alternative methods remains intact as data sets grow, indicating CYB's gains are a genuine improvement in learning efficiency, not an artifact of data saturation. This experiment finds further support for hypothesis **H1**, the superiority of CYB over TBYS, here in data efficiency.

Uniform-stopping ablation. CYB differs from TBYS in two respects: it incorporates learnable abstention and it trains the output token on each pause step. We have hypothesized that learned abstention is critical (**H3**). To support this hypothesis, we conducted an ablation in which learned abstention is eliminated. We imposed a uniform distribution over stopping step via the hyperparameter vector $\boldsymbol{\omega} = (0.25, 0.25, 0.25, 0.25)$ and trained a variant of CYB with a maximum of $M = 4$ steps, ignoring the <DON'T KNOW> output. This ablation that trains the correct response at each pause step performed extremely poorly: on Gemma3-4B, C4 evaluation perplexity is 18.60 for the ablation versus 10.13 for CYB. The ablated model did improve reliably on the training set over fine tuning, but showed the classic sign of overfitting on the evaluation set: perplexity dropped and then ramped back up; the lowest perplexity achieved was 13.67. Thus, we find evidence that allowing the model to control the processing flow with <DON'T KNOW> outputs drives performance gains, distinguishing CYB from adaptive-computation time or anytime-prediction methods.

Hyperparameter exploration. We explored two hyperparameters of CYB introduced in Section 2: $\boldsymbol{\omega}$ and γ_0. Instead of allowing the three-pause CYB model to use as many steps as it wished, we imposed an external stop time distribution. We use the notation $\boldsymbol{\omega} = \omega_1 : \omega_2 : \omega_3 : \omega_4$ to denote the ratio of probabilities of forcing the model to stop after processing steps 1-4. When CYB has full determination of when to stop within the pause sequence, $\boldsymbol{\omega} =$ 0:0:0:1. The other hyperparameter, γ_0, controls the exponential discounting of accuracy at each step (i.e., accuracy is discounted by γ_0^{i-1} at step i); $\gamma_0 = 1$ for no discounting.

Turning first to $\boldsymbol{\omega}$, Figure 5 shows perplexity for fine-tuned models with various hyperparameters and 3 pauses per token. The leftmost bar imposes no early stopping and no discounting—the result we previously presented. The rightmost two bars indicate performance when early stopping is imposed, either uniformly ($\boldsymbol{\omega} =$ 1:1:1:1) or with a bimodal distribution ($\boldsymbol{\omega} =$ 4:1:1:4). Consistent with **H3**, which posits that CYB's advantage over TBYS stems from having control over when it responds, we see that losing this control via early stopping severely harms its performance.

In contrast, when exponential discounting ($\gamma_0 = 0.99$) is applied, only a small cost in perplexity arises, as represented by the second bar from the left in Figure 5. Nonetheless, discounting is very effective in reducing the expected stopping step of the model, as indicated by the distribution shift between the first two columns of Figure 7. To explain these distributions, CYB's <DON'T KNOW> output allows us to compute the *latency* distribution (Equation 1), i.e., the distribution over which step

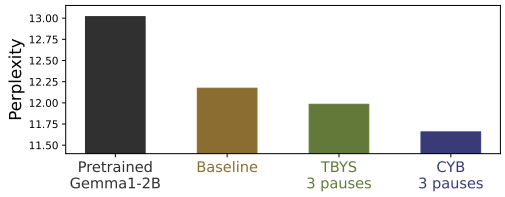

Figure 4: C4 evaluation set perplexity for Gemma1-2B architectures pretrained from scratch on C4 (except for the black bar, which uses open-source weights). Model variants include a baseline condition without pauses and models with three pauses per real token: TBYS [14] and CYB (ours).

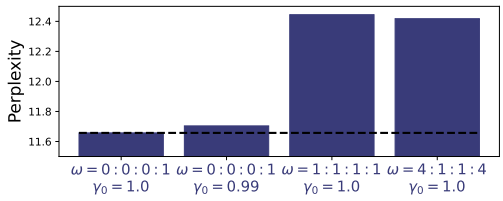

Figure 5: C4 evaluation set perplexity for various hyperparameters of CYB with Gemma1-2B fine-tuned on C4. Hyperparameter ω refers to the distribution of stopping times and is indicated by ratios for steps 1-4 (i.e., the initial input token and the following 3 pause tokens). Hyperparameter γ_0 refers to an exponential discount factor.

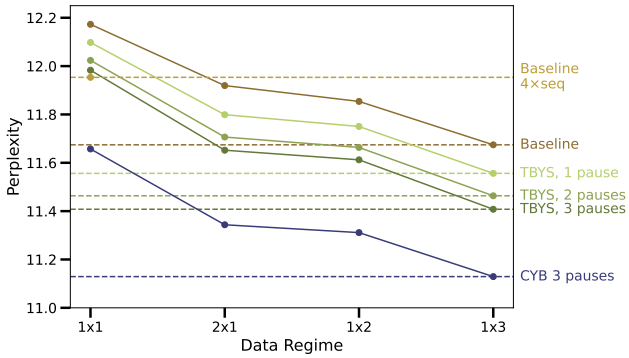

Figure 6: C4 evaluation set perplexity for various fine-tuned Gemma1-2B models based on a given data regime. 1×1 and 2×1 refer to 1 and 2 passes through the fine-tuning set, respectively. 1×2 and 1×3 refer to a single pass through 2 or 3 times the dataset size of 1×1. The dashed lines indicate the perplexity of each model in the 1×3 training regime, except for *Baseline* $4\times$*seq* where we only report performance in the 1×1 regime.

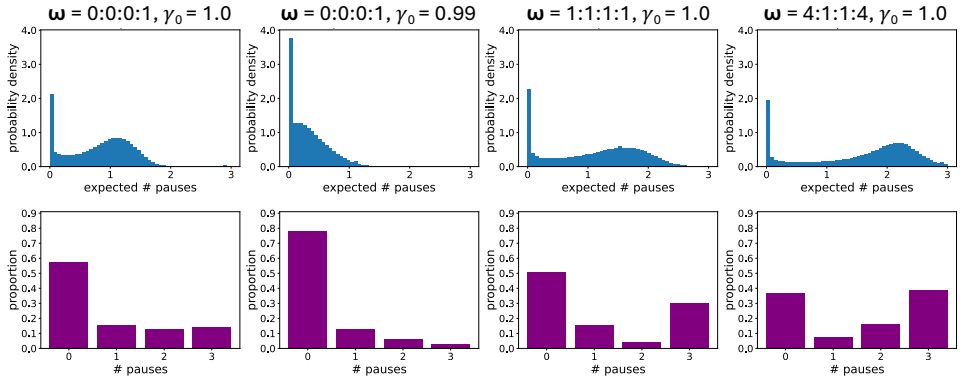

Figure 7: Latency distribution for four hyperparameter settings of CYB. Blue histograms show the distribution over evaluation tokens of the expected latency (the number of pause steps utilized). Purple bar graphs show the latency distribution averaged across tokens.

the model reads out its response. Figure 7 presents statistics of the latency distribution for the four hyperparameter settings of CYB previously described in Figure 5. The blue curves are a probability density over the expected latency for individual tokens. The purple bars show the latency distribution averaged over tokens. The distributions in the first column ($\gamma_0 = 1$) shift leftward in the second column ($\gamma_0 = .99$), consistent with hypothesis **H4** that we can manipulate the model's speed-accuracy trade-off with discounting. With discounting, delaying a response is penalized unless the delay results in a corresponding boost in accuracy.

In the leftmost column of Figure 7, the model has no discounting penalty ($\gamma_0 = 1$) and need not accommodate forced stops ($\boldsymbol{\omega} = 0{:}0{:}0{:}1$), yet CYB is nonetheless judicious in its use of pauses. This finding seems surprising given that additional pause steps should not *harm* performance. However, the finding is consistent with hypothesis **H3**, which asserts the advantage of CYB over TBYS is that CYB can stop when it is ready to respond instead of holding its response until the final step.

Paradoxically, forcing the model to stop early shifts its preferred latency distribution slower, as seen in the last two columns of Figure 7. Perhaps the increased likelihood of <DON'T KNOW> is simply a reflection of its worse performance (Figure 5). We explored using soft constraints (secondary losses) to encourage the model to adhere to a target stop-time distribution, as an alternative to discounting. However, this approach yielded strictly worse performance from CYB. See Appendix C for details.

Calibration of model confidence and accuracy. If successful, CYB should train models to self-calibrate (Hypothesis **H2**): <DON'T KNOW> should have high probability when a delay in responding will improve model output. Appendix E.1 provides a theoretical argument for this conjecture by showing that model gradients depend on improving accuracy over steps. Appendix E.2 presents empirical evidence that the model implicitly predicts its own accuracy at subsequent steps and is sensitive to this prediction in using the <DON'T KNOW> output. A statistically reliable positive correlation is found between <DON'T KNOW> probability and an accuracy gain with additional processing for all steps of a three-pause model, supporting the calibration hypothesis **H2**.

Table 1: Downstream evaluation top-1 accuracy (%). HellaSwag and MMLU are evaluated on C4 fine-tuned checkpoints. GSM8K results are means and standard errors over 10 replications in which we additionally fine tune on the GSM8K training set.

Method	**HellaSwag** (10-shot)	**MMLU** (5-shot)	**GSM8K** (8-shot)
Baseline	24.68	49.16	47.60
TBYS	24.74	52.70	53.16 [SEM 0.23]
CYB	**27.94**	**54.00**	**53.91** [SEM 0.30]

Table 2: Statistics of pause counts for selected tokens following CYB training

	Low Median	**High Median**
Low Variance	isn, wasn, didn, doesn, don, according, able, However, plenty, etc, Inc, &, non, addition	devices, players, students, projects, applications, challenges, environment, systems, stories, games, groups, events, patients, families, teams
High Variance	g, e, to, entry, S, won, co, New, Y, V, F, Z, World, of, Air	materials, images, @, site, pictures, photos, sites, code, College, wood, page, file, professionals, is, homes

Downstream task performance. We evaluate Gemma3-4B models fine-tuned on C4 with three pauses per token (zero for the baseline) on several downstream benchmarks: HellaSwag 10-shot [51], MMLU 5-shot [18], and GSM8K 8-shot Chain of Thought (CoT) [8], reporting top-one accuracy in all cases. In these experiments, we insert three pauses after every real token for CYB and TBYS, including tokens in the prompt and in the response, at both fine-tuning and evaluation. For CYB, we generate responses by deterministic selection of the most probable response token, including <DON'T KNOW>. This procedure means that we run the sequential-decision framework selecting the most probable choice at every stage. For GSM8K, the C4 fine-tuned models achieved 0% accuracy in a zero-shot setting; we therefore additionally fine-tuned each model on the GSM8K training set for 2 epochs with a learning rate of $10^{-4}/\sqrt{d_{\text{model}}}$ and a warmup with 18 steps. We conducted ten replications with different random seeds which determined batch composition and ordering. Results are summarized in Table 1 and match results of the perplexity analysis: CYB yields a consistent improvement in accuracy over TBYS, and both beat the baseline no-pause model. For the comparison of TBYS and CYB on GSM8K, we observe a reliable improvement in accuracy with CYB over TBYS (two-tailed paired t-test: $t(9) = 2.85$, $p = 0.019$).

Qualitative results. Table 2 shows pause-count statistics for selected tokens for the three-pause CYB model. The Table presents tokens which tend to be followed by few (*low median*) or many (*high median*) pauses, and those which tend to have the same number of pauses regardless of their context (*low variance*) or have context-sensitive pause durations (*high variance*). There are clear regularities in these tokens, such as the upper right cell being plural nouns, the upper left cell having highly predictable completions, and the lower left cell containing single letters.

Figure 1b presents a sample evaluation token sequence. Each token's background is colored to indicate the expected number of pause steps requested by the model in order to predict that token. (That is, the coloring reflects the processing steps conditioned on the preceding token.) The coloring ranges from white to dark blue, corresponding to zero pause steps (1 total step) to 3 pause steps (4 total steps). The most obvious feature of the pause pattern is that it is not uniform but varies on a token-by-token basis. For example, the pause following the token 'traditional' in Figure 1b tends to be long; in contrast, the model does not slow down when it is predicting a string of digits. More examples of shaded text can be found in Supplementary Figures F.6 and F.7.

5 Discussion

We introduced Catch Your Breath (CYB), a sequential-decision framework and loss function that supports inference-time scaling of per-token computation. CYB has the option of utilizing additional compute steps when the output is uncertain, allowing it to pause and delay its output. In classical classification with abstention, epistemic uncertainty stems from data scarcity or gaps in

training coverage. Under the CYB objective, however, uncertainty is fundamentally a reflection of computational constraints: the model is uncertain not because it lacks representation in its weights, but because it has not yet performed sufficient computations to integrate long-range context. As our calibration metrics show, subsequent pause steps allow the model to resolve this internal uncertainty using test-time compute.

We described CYB as performing *width-based* scaling, which is an underexplored approach relative to *depth-based* scaling (e.g., looped transformers). While both width-based and depth-based scaling of computation can be performed without an increase in the memory footprint of the model weights, width-based scaling has a much lighter impact on inference latency. The two forms of scaling can be combined, though, and extend model expressivity in complementary ways [31].

The existing paradigm used for width-based scaling, which we refer to as Think Before You Speak (TBYS), is a limiting case of CYB suffering from the fact that it lacks autonomy over the final readout point. We presented strong evidence that CYB outperforms TBYS (hypothesis **H1**), and that CYB's benefit stems from its control over when to respond (hypothesis **H3**). Our focus on comparing CYB and TBYS allows for a strict, rigorously matched comparison: the two methods have identical memory and computation requirements at both training and inference; they differ only in the loss function used and the readout point. The benefits for CYB over TBYS are essentially cost free.

Although we demonstrated that CYB can be used to control the speed-accuracy trade-off (hypothesis **H4**), it is important to emphasize that CYB is not motivated by raw inference-time compute savings; rather, it is about providing the architecture with flexible, context-dependent control over its computation. Just as human readers selectively modulate fixation durations to resolve lexical ambiguities or integrate complex discourse (see Appendix A), CYB-trained models naturally learn to pause only when the local transition dynamics demand it. This behavior is conceptually aligned with looped transformers that utilize internal routing components to decide whether or not to continue depth recurrence. However, width-based adaptation through CYB offers distinct structural and engineering advantages over these depth-recurrent models, including simplicity and elegance of the routing rule, parallelizability, and formal expressivity.

Limitations and future directions. The recipe we used for variable computation in CYB involves inserting the maximum number of pauses after each real token. While this recipe is highly parallelizable and introduces no computational or memory overhead *relative to TBYS*, it can limit the effective sequence length for long-form generation. One might imagine that running the model stochastically—inserting a <PAUSE> only when an explicit <DON'T KNOW> is triggered—would be interesting, but the only significant benefit of stochastic rollouts would be in KV cache memory, given that multiple pauses can be inserted and computed in parallel. Recent work suggests an alternative approach to reducing the memory footprint. In a particular setting, Monea et al. [32] and Langford et al. [26] found that benefits of pausing can be obtained without storing all pause steps in KV cache. This approach ought to be tested for CYB.

Other potential directions for future research include:

- *Non-causal processing of the pauses.* Since <PAUSE> tokens are placed in the input stream simultaneously, they can inform one another as activation is propagated forward. Then the CYB loss trains the model which <PAUSE> token output to use for its response.
- *Selective insertion of pauses.* Instead of placing a sequence of pauses after every real token, one could instead selectively insert the pauses at positions most likely to benefit from them, as determined by analysis of trained models (e.g., Table 2).
- *Variable pause budgets.* Our work primarily inserted two or three pauses after each input token, both at training and evaluation. There is no need for the training and generation budgets to be the same. Indeed, Goyal et al. [14] observed benefit of inserting a larger number of pauses at generation time than the model had seen during training.
- *Looping.* We argued that looped transformers and pausing—depth-based and width-based scaling—were independent architectural decisions and are motivated by different concerns. Nonetheless, it would be valuable to study whether interactions exist; e.g., pausing may be more effective in the context of looped models.
- *Range of models and model scales.* We tested only the Gemma family of models and primarily smaller models. CYB should be evaluated on other families, larger architectures, and MoE-based transformers.

Acknowledgments and Disclosure of Funding

We are grateful to the insights and participation of Jasper Uijlings early in the course of this research, and to the anonymous reviewers and area chair who provided constructive feedback on the research and manuscript. This research was supported by Google.

References

[1] Alabdulmohsin, I. and Zhai, X. (2025). Recursive inference scaling: A winning path to scalable inference in language and multimodal systems. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[2] Arnon, T. and Lavidor, M. (2023). Cognitive control in processing ambiguous idioms: evidence from a self-paced reading study. *Journal of Psycholinguistic Research*, 52(1):261–281.

[3] Bae, S., Kim, Y., Bayat, R., Kim, S., Ha, J., Schuster, T., Fisch, A., Harutyunyan, H., Ji, Z., Courville, A., and Yun, S.-Y. (2025). Mixture-of-recursions: Learning dynamic recursive depths for adaptive token-level computation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

[4] Banino, A., Balaguer, J., and Blundell, C. (2021). PonderNet: Learning to ponder. In *8th ICML Workshop on Automated Machine Learning (AutoML)*.

[5] Bartlett, P. L. and Wegkamp, M. H. (2008). Classification with a reject option using a hinge loss. *Journal of Machine Learning Research*, 9(59):1823–1840.

[6] Burtsev, M. S., Kuratov, Y., Peganov, A., and Sapunov, G. V. (2021). Memory transformer. arXiv:2006.11527 [cs.CL].

[7] Chen, Y., Shang, J., Zhang, Z., Xie, Y., Sheng, J., Liu, T., Wang, S., Sun, Y., Wu, H., and Wang, H. (2025). Inner thinking transformer: Leveraging dynamic depth scaling to foster adaptive internal thinking. arXiv:2502.13842 [cs.CL].

[8] Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. (2021). Training verifiers to solve math word problems. arXiv:2110.14168 [cs.LG].

[9] Cohen, R., Dobler, K., Biran, E., and de Melo, G. (2024). I don't know: Explicit modeling of uncertainty with an [IDK] token. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[10] Darcet, T., Oquab, M., Mairal, J., and Bojanowski, P. (2024). Vision transformers need registers. arXiv:2309.16588 [cs.CV].

[11] Dong, Q., Dong, L., Tang, Y., Ye, T., Sun, Y., Sui, Z., and Wei, F. (2025). Reinforcement pre-training. arXiv:2506.08007 [cs.CL].

[12] Geiping, J., McLeish, S., Jain, N., Kirchenbauer, J., Singh, S., Bartoldson, B. R., Kailkhura, B., Bhatele, A., and Goldstein, T. (2025). Scaling up test-time compute with latent reasoning: A recurrent depth approach. arXiv:2502.05171 [cs.LG].

[13] Giannou, A., Rajput, S., Sohn, J.-Y., Lee, K., Lee, J. D., and Papailiopoulos, D. (2023). Looped transformers as programmable computers. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 11398–11442. PMLR.

[14] Goyal, S., Ji, Z., Rawat, A. S., Menon, A. K., Kumar, S., and Nagarajan, V. (2024). Think before you speak: Training language models with pause tokens. In *ICLR*.

[15] Graves, A. (2016). Adaptive computation time for recurrent neural networks. arXiv:1603.08983.

[16] Hao, S., Sukhbaatar, S., Su, D., Li, X., Hu, Z., Weston, J., and Tian, Y. (2024). Training large language models to reason in a continuous latent space. arXiv:2412.06769 [cs.CL], also COLM 2025.

[17] Hendrickx, K., Perini, L., Van der Plas, D., Meert, W., and Davis, J. (2024). Machine learning with a reject option: A survey. *Machine Learning*, 113(5):3073–3110.

[18] Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. (2020). Measuring massive multitask language understanding. *CoRR*, abs/2009.03300.

[19] Herel, D. and Mikolov, T. (2023). Thinking tokens for language modeling (extended abstract). In *Eighth Conference on Artificial Intelligence and Theorem Proving (AITP 2023)*, Aussois, France.

[20] Iuzzolino, M. L., Mozer, M. C., and Bengio, S. (2021). Improving anytime prediction with parallel cascaded networks and a temporal-difference loss. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*.

[21] Jeddi, A., Ciccone, M., and Taati, B. (2026). Loopformer: Elastic-depth looped transformers for latent reasoning via shortcut modulation. In *The Fourteenth International Conference on Learning Representations*.

[22] Just, M. A. and Carpenter, P. A. (1980). A theory of reading: From eye fixations to comprehension. *Psychological Review*, 87(4):329–354.

[23] Kim, E., Kim, S., and Thorne, J. (2025). Learning to insert [pause] tokens for better reasoning. arXiv:2506.03616 [cs.CL].

[24] Koishekenov, Y., Lipani, A., and Cancedda, N. (2025). Encode, think, decode: Scaling test-time reasoning with recursive latent thoughts. arXiv:2510.07358 [cs.LG].

[25] Lanchantin, J., Toshniwal, S., Weston, J., Szlam, A., and Sukhbaatar, S. (2023). Learning to reason and memorize with self-notes. arXiv:2305.00833 [cs.LG].

[26] Langford, J., Godey, N., Monea, G., Artzi, Y., Dong, H., Fan, Y., de Rosa, G., and Zhan, Z. (2026). Almost free state prediction separation. arXiv:2609.03807 [cs.LG].

[27] Li, Z., Li, Y., and Zhou, T. (2025). Skip a layer or loop it? Test-time depth adaptation of pretrained LLMs. arXiv:2507.07996 [cs.LG].

[28] Liu, Z., Wang, Z., Liang, P. P., Salakhutdinov, R. R., Morency, L.-P., and Ueda, M. (2019). Deep gamblers: Learning to abstain with portfolio theory. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

[29] London, C. and Kanade, V. (2025). Pause tokens strictly increase the expressivity of constant-depth transformers. arXiv:2505.21024 [cs.LG].

[30] McLeish, S., Li, A., Kirchenbauer, J., Kalra, D. S., Bartoldson, B. R., Kailkhura, B., Schwarzschild, A., Geiping, J., Goldstein, T., and Goldblum, M. (2025). Teaching pretrained language models to think deeper with retrofitted recurrence. arXiv:2511.07384 [cs.CL].

[31] Merrill, W. and Sabharwal, A. (2025). A little depth goes a long way: The expressive power of log-depth transformers. arXiv:2503.03961 [cs.LG].

[32] Monea, G., Godey, N., Brantley, K., and Artzi, Y. (2026). The state-prediction separation hypothesis. arXiv:2607.01218 [cs.CL].

[33] Ng, D. N. (2026). LLM neuroanatomy: How I topped the LLM leaderboard without changing a single weight. https://dnhkng.github.io/posts/rys/.

[34] Nowak, A. I., Mercea, O.-B., Arnab, A., Pfeiffer, J., Dauphin, Y., and Evci, U. (2024). Towards optimal adapter placement for efficient transfer learning. arXiv:2410.15858 [cs.LG].

[35] Oh, B.-D. and Schuler, W. (2022). Entropy- and distance-based predictors from GPT-2 attention patterns predict reading times over and above GPT-2 surprisal. In Goldberg, Y., Kozareva, Z., and Zhang, Y., editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9324–9334, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

[36] Pfau, J., Merrill, W., and Bowman, S. R. (2024). Let's think dot by dot: Hidden computation in transformer language models. In *First Conference on Language Modeling*.

[37] Phan, D., Hoffman, M. D., Dohan, D., Douglas, S., Le, T. A., Parisi, A., Sountsov, P., Sutton, C., Vikram, S., and Saurous, R. A. (2023). Training chain-of-thought via latent-variable inference. arXiv:2312.02179 [cs.LG].

[38] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

[39] Rodkin, I., Orel, D., Smirnov, K., Bolatov, A., Elbouardi, B., Hassan, B., Kuratov, Y., Bulatov, A., Nakov, P., Baldwin, T., Shelmanov, A., and Burtsev, M. (2025). Beyond memorization: Extending reasoning depth with recurrence, memory and test-time compute scaling. arXiv:2508.16745 [cs.LG].

[40] Ruan, Y., Band, N., Maddison, C. J., and Hashimoto, T. (2025). Reasoning to learn from latent thoughts. arXiv:2503.18866 [cs.LG].

[41] Stewart, A. J., Kidd, E., and Haigh, M. (2009). Early sensitivity to discourse-level anomalies: Evidence from self-paced reading. *Discourse Processes*, 46:46–69.

[42] Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. (2023). Roformer: Enhanced transformer with rotary position embedding. arXiv:2104.09864 [cs.CL].

[43] Team, G., Kamath, A., Ferret, J., Pathak, S., Vieillard, N., Merhej, R., Perrin, S., Matejovicova, T., Ramé, A., Rivière, M., Rouillard, L., Mesnard, T., Cideron, G., bastien Grill, J., Ramos, S., Yvinec, E., Casbon, M., Pot, E., Penchev, I., Liu, G., Visin, F., Kenealy, K., Beyer, L., Zhai, X., Tsitsulin, A., Busa-Fekete, R., Feng, A., Sachdeva, N., Coleman, B., Gao, Y., Mustafa, B., Barr, I., Parisotto, E., Tian, D., Eyal, M., Cherry, C., Peter, J.-T., Sinopalnikov, D., Bhupatiraju, S., Agarwal, R., Kazemi, M., Malkin, D., Kumar, R., Vilar, D., Brusilovsky, I., Luo, J., Steiner, A., Friesen, A., Sharma, A., Sharma, A., Gilady, A. M., Goedeckemeyer, A., Saade, A., Feng, A., Kolesnikov, A., Bendebury, A., Abdagic, A., Vadi, A., György, A., Pinto, A. S., Das, A., Bapna, A., Miech, A., Yang, A., Paterson, A., Shenoy, A., Chakrabarti, A., Piot, B., Wu, B., Shahriari, B., Petrini, B., Chen, C., Lan, C. L., Choquette-Choo, C. A., Carey, C., Brick, C., Deutsch, D., Eisenbud, D., Cattle, D., Cheng, D., Paparas, D., Sreepathihalli, D. S., Reid, D., Tran, D., Zelle, D., Noland, E., Huizenga, E., Kharitonov, E., Liu, F., Amirkhanyan, G., Cameron, G., Hashemi, H., Klimczak-Plucińska, H., Singh, H., Mehta, H., Lehri, H. T., Hazimeh, H., Ballantyne, I., Szpektor, I., Nardini, I., Pouget-Abadie, J., Chan, J., Stanton, J., Wieting, J., Lai, J., Orbay, J., Fernandez, J., Newlan, J., yeong Ji, J., Singh, J., Black, K., Yu, K., Hui, K., Vodrahalli, K., Greff, K., Qiu, L., Valentine, M., Coelho, M., Ritter, M., Hoffman, M., Watson, M., Chaturvedi, M., Moynihan, M., Ma, M., Babar, N., Noy, N., Byrd, N., Roy, N., Momchev, N., Chauhan, N., Sachdeva, N., Bunyan, O., Botarda, P., Caron, P., Rubenstein, P. K., Culliton, P., Schmid, P., Sessa, P. G., Xu, P., Stanczyk, P., Tafti, P., Shivanna, R., Wu, R., Pan, R., Rokni, R., Willoughby, R., Vallu, R., Mullins, R., Jerome, S., Smoot, S., Girgin, S., Iqbal, S., Reddy, S., Sheth, S., Põder, S., Bhatnagar, S., Panyam, S. R., Eiger, S., Zhang, S., Liu, T., Yacovone, T., Liechty, T., Kalra, U., Evci, U., Misra, V., Roseberry, V., Feinberg, V., Kolesnikov, V., Han, W., Kwon, W., Chen, X., Chow, Y., Zhu, Y., Wei, Z., Egyed, Z., Cotruta, V., Giang, M., Kirk, P., Rao, A., Black, K., Babar, N., Lo, J., Moreira, E., Martins, L. G., Sanseviero, O., Gonzalez, L., Gleicher, Z., Warkentin, T., Mirrokni, V., Senter, E., Collins, E., Barral, J., Ghahramani, Z., Hadsell, R., Matias, Y., Sculley, D., Petrov, S., Fiedel, N., Shazeer, N., Vinyals, O., Dean, J., Hassabis, D., Kavukcuoglu, K., Farabet, C., Buchatskaya, E., Alayrac, J.-B., Anil, R., Dmitry, Lepikhin, Borgeaud, S., Bachem, O., Joulin, A., Andreev, A., Hardin, C., Dadashi, R., and Hussenot, L. (2025). Gemma 3 technical report. arXiv:2503.19786 [cs.CL].

[44] Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., Tafti, P., Hussenot, L., Sessa, P. G., Chowdhery, A., Roberts, A., Barua, A., Botev, A., Castro-Ros, A., Slone, A., Héliou, A., Tacchetti, A., Bulanova, A., Paterson, A., Tsai, B., Shahriari, B., Lan, C. L., Choquette-Choo, C. A., Crepy, C., Cer, D., Ippolito, D., Reid, D., Buchatskaya, E., Ni, E., Noland, E., Yan, G., Tucker, G., Muraru, G.-C., Rozhdestvenskiy, G., Michalewski, H., Tenney, I., Grishchenko, I., Austin, J., Keeling, J., Labanowski, J., Lespiau, J.-B., Stanway, J., Brennan, J., Chen, J., Ferret, J., Chiu, J., Mao-Jones, J., Lee, K., Yu, K., Millican, K.,

Sjoesund, L. L., Lee, L., Dixon, L., Reid, M., Mikuła, M., Wirth, M., Sharman, M., Chinaev, N., Thain, N., Bachem, O., Chang, O., Wahltinez, O., Bailey, P., Michel, P., Yotov, P., Chaabouni, R., Comanescu, R., Jana, R., Anil, R., McIlroy, R., Liu, R., Mullins, R., Smith, S. L., Borgeaud, S., Girgin, S., Douglas, S., Pandya, S., Shakeri, S., De, S., Klimenko, T., Hennigan, T., Feinberg, V., Stokowiec, W., hui Chen, Y., Ahmed, Z., Gong, Z., Warkentin, T., Peran, L., Giang, M., Farabet, C., Vinyals, O., Dean, J., Kavukcuoglu, K., Hassabis, D., Ghahramani, Z., Eck, D., Barral, J., Pereira, F., Collins, E., Joulin, A., Fiedel, N., Senter, E., Andreev, A., and Kenealy, K. (2024). Gemma: Open models based on gemini research and technology. arXiv:2403.08295 [cs.CL].

[45] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. (2023). Chain-of-thought prompting elicits reasoning in large language models. arXiv:2201.11903 [cs.CL].

[46] Williams, C. K. I. (2026). A review of and best practices for comparing machine learning models. Unpublished manuscript, dated 12 March 2026; Retrieved from https://homepages.inf.ed.ac.uk/ckiw/postscript/compare_models_120326.pdf.

[47] Yang, L., Lee, K., Nowak, R. D., and Papailiopoulos, D. (2024). Looped transformers are better at learning learning algorithms. In *The Twelfth International Conference on Learning Representations*.

[48] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. arXiv:2210.03629 [cs.CL].

[49] Yu, Z., Belinkov, Y., and Ananiadou, S. (2025). Back attention: Understanding and enhancing multi-hop reasoning in large language models. In Christodoulopoulos, C., Chakraborty, T., Rose, C., and Peng, V., editors, *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 11257–11272, Suzhou, China. Association for Computational Linguistics.

[50] Zelikman, E., Wu, Y., Mu, J., and Goodman, N. D. (2022). STaR: Bootstrapping reasoning with reasoning. arXiv:2203.14465 [cs.LG].

[51] Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. (2019). HellaSwag: Can a machine really finish your sentence? *CoRR*, abs/1905.07830.

[52] Zellner, A. (1988). Optimal information processing and Bayes's theorem. *The American Statistician*, 42(4):278–280.

[53] Zeng, B., Song, S., Huang, S., Wang, Y., Li, H., He, Z., Wang, X., Li, Z., and Lin, Z. (2026). PonderLM: Pretraining language models to ponder in continuous space. In *The Fourteenth International Conference on Learning Representations*.

[54] Zhu, R.-J., Wang, Z., Hua, K., Zhang, T., Li, Z., Que, H., Wei, B., Wen, Z., Yin, F., Xing, H., Li, L., Shi, J., Ma, K., Li, S., Kergan, T., Smith, A., Qu, X., Hui, M., Wu, B., Min, Q., Huang, H., Zhou, X., Ye, W., Liu, J., Yang, J., Shi, Y., Lin, C., Zhao, E., Cai, T., Zhang, G., Huang, W., Bengio, Y., and Eshraghian, J. (2025). Scaling latent reasoning via looped language models. arXiv:2510.25741 [cs.LG].

A How people read

The inspiration for this research comes from studies of human reading. When gaze is tracked as an individual reads, inter-saccade durations are highly nonuniform. Readers fixate longer when processing demands rise, e.g., at points where information can be integrated across phrases, where inferences can be made at the end of sentences, and where low-frequency terms occur [22]. Figure A.1 shows fixation order within a sentence for a human reader as well as the fixation duration. Note that short function words are skipped or quickly glanced over. When an uncommon interpretation of a word or sentence is required, readers pause. For instance, when idioms that typically have a figurative meaning (e.g., 'break the ice') are used in a literal way, processing time slows at the point where the disambiguating information is provided [2]. Similarly, readers pause when anomalous or unexpected information is provided, including attributes and spatial position of a character or object [41]. These various results suggest that people integrate information in an ongoing manner, where each incoming word and clause is processed with respect to prior context, and reading time is governed by the computational demands of online evaluative processes.

In humans, the ongoing, word-by-word integration processes might be described as *microinference*, in contrast to *macroinference*, which involves solving complex reasoning and planning problems. In AI, past research on pause tokens has focused on their utility for macroinference [14, 19, 23, 36]. Our present work is aimed at emphasizing the role of delays for microinference, motivated by the need to integrate and interpret information in an ongoing fashion. One important clue that this focus will be productive comes from a study which found that GPT-2 attention patterns could predict human reading times, in particular measures of the dispersion of attention and the change in attention patterns across time steps [35].

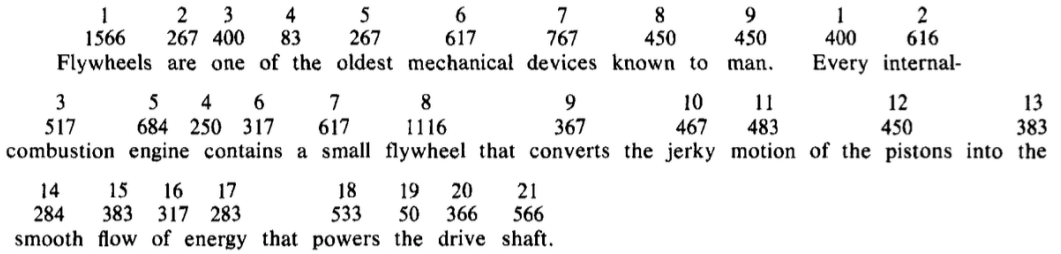

Figure A.1: Gaze fixations of a human reader. Indices indicate order of gaze within a sentence along with fixation duration (in msec). Reprinted from Just and Carpenter (1980, Figure 1).

B Notation used in the main article

Table B.1: Notation Used in the Main Article

Term	Meaning
M	maximum number of steps granted by world for any token
S	r.v. denoting # steps at which model generates result (either by world or model <DON'T KNOW>), $1 \leq S \leq M$
ω_i	prob. that world terminates processing at step i
d_i	prob. of model abstention (<DON'T KNOW>) at step i
s_i	prob. of outputting result (self- or world-induced) at step i; shorthand for $\Pr(S = i)$
t_i	prob. that the model selects ground-truth token at step i
γ_i	discount factor on accuracy at step i, $0 \leq \gamma_i \leq 1$
γ_0	hyperparameter for exponential discounting, with $\gamma_i = \gamma_0^{i-1}$
ρ_i	[CYB-VA, CYB-DP] prior prob. of self-induced termination at step i
α	[CYB-DP] penalty coefficient on prior-distribution mismatch

C Guiding CYB to obtain a desired stop-time distribution

We describe two variants of CYB that impose an additional constraint on model outputs with the goal of obtaining a desired stop-time distribution. We refer to this target distribution as $\boldsymbol{\rho}$.

C.1 Variational approach (CYB-VA)

This approach aims to obtain a desired distribution $\boldsymbol{\rho}$ and to minimize the loss with respect to this distribution, i.e.,

$$\ell_{\text{CYB}\rho} = -\log \mathbb{E}_{i\sim\text{Cat}(\boldsymbol{\rho})} \left[\gamma_i t_i\right].$$

However, this loss does not depend on $\boldsymbol{d}$ and will thus fail to train the <DON'T KNOW> output, let alone train it to be consistent with $\boldsymbol{\rho}$. To achieve this goal, we take a variational approach and train the model using the ELBO. Treating the model's likelihood at step i as the penalized value $\Pr[\text{target}|S = i] = \gamma_i t_i$, the ELBO is the right hand side of the inequality:

$$\log \mathbb{E}_{i\sim\text{Cat}(\boldsymbol{\rho})} \left[\gamma_i t_i\right] \geq \mathbb{E}_{i\sim\text{Cat}(\boldsymbol{s})} \log \left[\gamma_i t_i\right] - D_{\text{KL}}(\boldsymbol{s} \,||\, \boldsymbol{\rho})$$

where we assume as before that externally induced stops are eliminated. The <DON'T KNOW> distribution, $\boldsymbol{d}$, is incorporated via the stopping time distribution $\boldsymbol{s} \equiv \Pr(S|\boldsymbol{d})$ (Equation 1). We then define a negative ELBO loss,

$$\ell_{\text{CYB-VA}} = -\mathbb{E}_{i\sim\text{Cat}(\boldsymbol{s})} \log \left[\gamma_i t_i\right] + D_{\text{KL}}(\boldsymbol{s} \,||\, \boldsymbol{\rho}) \,.$$

Minimizing $\ell_{\text{CYB-VA}}$, as an upper bound on our desired loss $\ell_{\text{CYB}\rho}$, trains the model's predictions ($\boldsymbol{t}$) by optimizing the expected discounted likelihood under $\boldsymbol{s}$. It also trains the model's stopping time distribution ($\boldsymbol{s}$) toward the posterior, following the optimization view of Bayesian inference [e.g., 52]. We can determine the equilibrium self-stop distribution by taking the gradient of the variational loss,

$$\nabla_{\boldsymbol{s}}\, \ell_{\text{CYB-VA}} = -\log \boldsymbol{\gamma} - \log \boldsymbol{t} - \log \boldsymbol{\rho} + \log \boldsymbol{s} + \mathbf{1} \,.$$

Introducing a Lagrange multiplier for the constraint $\sum_i s_i = 1$ yields the optimality condition $\log \boldsymbol{s} = \log \boldsymbol{\gamma} + \log \boldsymbol{t} + \log \boldsymbol{\rho} - (\lambda + 1)\mathbf{1}$, resulting in $s_i \propto \rho_i \gamma_i t_i$. The model will thus learn to stop at times when both the stopping prior and its discounted accuracy are high. This analysis also indicates that the prior, $\boldsymbol{\rho}$, and the discount factors, $\boldsymbol{\gamma}$, are redundant in CYB-VA. We thus assume $\gamma_i = 1$ in all experiments with CYB-VA. CYB-VA thus has hyperparameters $\boldsymbol{\rho}$, in addition to specifying M, the upper limit on the number of steps.

C.2 Distributional penalty (CYB-DP)

A final approach is to optimize for our original loss, $\mathcal{L}_{\text{CYB}}$ (Equation 3), and impose a penalty based on the deviation of the model's desired and observed stop-time distributions:

$$\ell_{\text{CYB-DP}} = \ell_{\text{CYB}} + \alpha\, D_{\text{KL}}(\boldsymbol{\rho} \,||\, \boldsymbol{s}) \,,$$

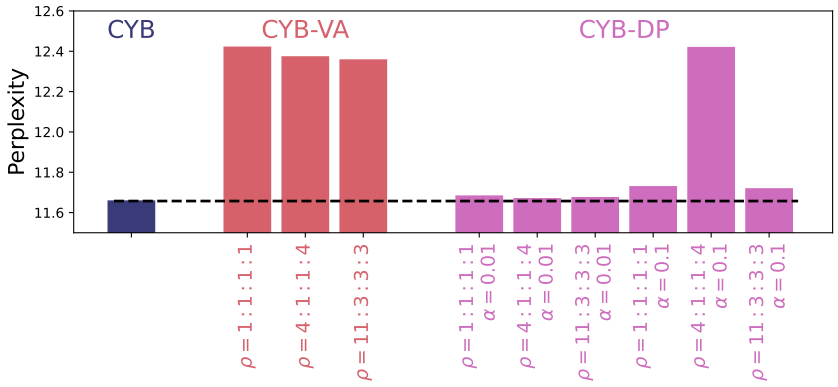

Figure C.2: C4 evaluation set perplexity for various hyperparameter settings of CYB and two variants, CYB-VA and CYB-DP, on Gemma1-2B fine-tuning experiment. The black dashed line indicates the perplexity of CYB. Hyperparameters ρ refer to the distribution of stopping times and are indicated by ratios for steps 1-4 (i.e., the initial input token and the following 3 pause tokens). Hyperparameter α is a penalty term associated with CYB-DP. See Table B.1 for notation.

where α specifies a penalty on the distribution mismatch. Note that the direction of the KL terms is reversed for $\ell_{\text{CYB-DP}}$ versus $\ell_{\text{CYB-VA}}$, in order to strongly penalize stopping times that are outside the support of the prior. (And we wish to allow the prior to have zero values.) CYB-DP has hyperparameters $\boldsymbol{\rho}$ and α, in addition to specifying M, the upper limit on the number of steps.

C.3 Results from CYB-VA and CYB-DP simulations

Perplexity. Figure C.2 shows perplexity for CYB, three hyperparameter settings for CYB-VA and six for CYB-DP on Gemma1-2B fine-tuning experiment. CYB-VA performs poorly for three different target stop-time distributions, ρ. We considered incorporating a scaling coefficient on the KL term, but CYB-VA's elegance stems from the fact that such a coefficient should not be required. For CYB-DP, which penalizes deviation from a target stop-time distribution, we tested two penalty coefficients, α, and three target distributions, ρ. The smaller penalty yielded better performance, and the similar performance over ρ suggests that the penalty had little impact. If the penalty is not shaping learning, CYB-DP becomes equivalent to CYB with no world-induced early stopping.

Distributions of stopping times. Stopping-time distributions for CYB-DP and CYB-VA are presented in Figures C.3 and C.4. These Figures are analogous to Figure 7 from the main paper. CYB-VA adheres to the prior distribution and in general shows little variance indicating that it is insensitive to the specific token and context, which is a bad sign for an adaptive-pause method. CYB-DP for large $\alpha = 0.1$ behaves closer to CYB-VA and for smaller $\alpha = 0.01$ prefers longer latencies than CYB. However, longer latencies do not result in a lower perplexity compared to CYB.

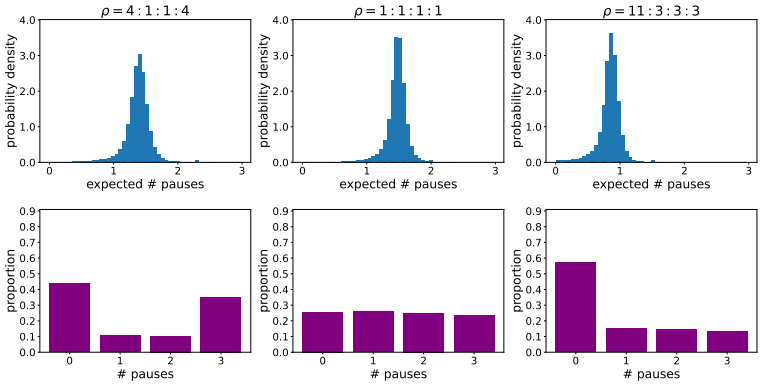

Figure C.3: Distribution of stopping times for CYB-VA. Blue histograms show the distribution over evaluation tokens of the expected number of pause steps. Purple bar graphs show the distribution over the number of pause steps across tokens.

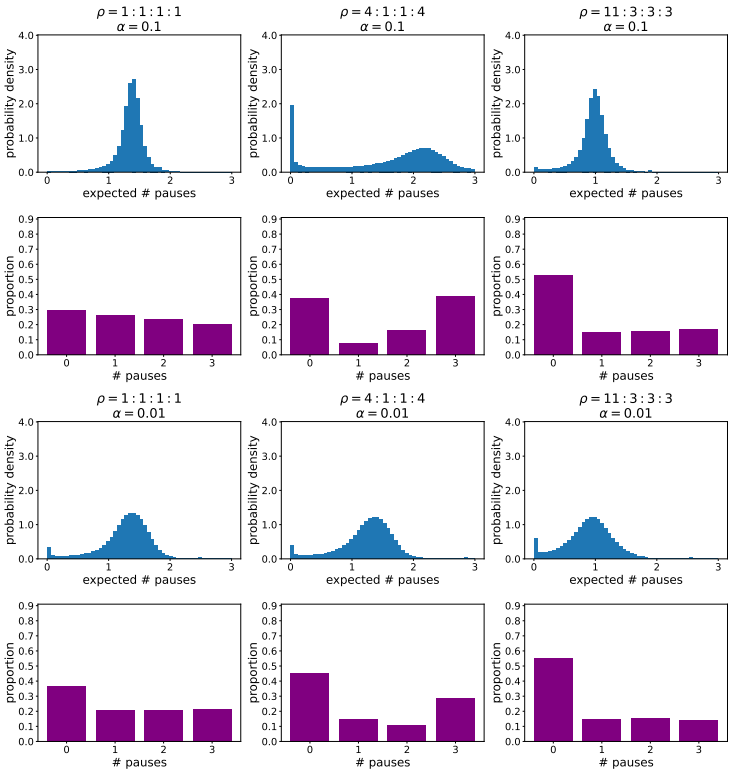

Figure C.4: Distribution of stopping times for CYB-DP with $\alpha = 0.1$ and $\alpha = 0.01$. Blue histograms show the distribution over evaluation tokens of the expected number of pause steps. Purple bar graphs show the distribution over the number of pause steps across tokens.

D Additional methodology

D.1 Training hyperparameters and data

For all the fine-tuning experiments, we fine-tuned either Gemma1-2B or Gemma3-4B models on a subset of 6.4 million sequences of the C4 dataset. For Gemma1-2B model we used batch size of 256, leading to a total of $25k$ iterations, while for Gemma3-4B model we used batch size of 64 leading to a total of 100k iterations. For pretraining experiments, we trained from scratch Gemma1-2B with batch size of 256 on 56×6.4 million sequences of the C4 dataset. For the evaluation, we use 160K sequences (59.8M tokens) from the C4 validation set.

We use a "concatenate then split" packing algorithm and a cosine learning rate decay with a linear warmup. We select a learning rate multiplier η_0 and obtain a maximum learning rate by dividing η_0 by the square root of the model embedding dimension. The minimum learning rate is given by multiplying the maximum learning rate by 10^{-2}. Before running all the experiments, we ran a sweep over $\eta_0 \in [10^{-2}, 5 \times 10^{-3}, 10^{-3}, 5 \times 10^{-4}, 5 \times 10^{-5}, 10^{-5}, 10^{-6}]$ where we ran the fine-tuning experiment on Gemma1-2B with baseline, TBYS with 3 pauses and CYB with 3 pauses. We found that $\eta_0 = 10^{-4}$ gave overall the best results for all the methods and this is what we used.

All the experiments are conducted on TPUv5 hardware.

D.2 Initializing <DON'T KNOW>

Because the pretrained model was never rewarded for assigning probability mass to the <DON'T KNOW> token, we wanted to ensure that the model actually had the potential to output <DON'T KNOW> when being fine-tuned with the CYB loss. To achieve this objective, model posteriors were renormalized to reflect a high prior for <DON'T KNOW>. If $\boldsymbol{\psi}$ is the prior distribution over tokens during pretraining and $\boldsymbol{\psi}'$ is the prior we wish to be reflected in the model posteriors, we add $\log(\boldsymbol{\psi}'/\boldsymbol{\psi})$ to the model logit vector. With uniform pretraining priors, i.e., $\psi_i = 1/|V|$ with V being the vocabulary, we set ψ'_{DK} for the <DON'T KNOW> and $\psi'_i = (1 - \psi'_{\text{DK}})/(|V| - 1)$ for all other indices i. In the reported experiments, we use $\psi'_{\text{DK}} = 0.9$. We found the models trained better with $\psi'_{\text{DK}} \geq 0.9$.

Comparison models. Our primary aim is to argue that CYB outperforms the memory- and compute-matched TBYS in its ability to utilize <PAUSE> tokens. TBYS is identical but is trained by the ordinary cross-entropy loss. For TBYS, we used the same position coding scheme as in the CYB models, even though the original TBYS model treated pauses like any other token for the purpose of position coding. (The original TBYS model also does not encode the first, second, etc. pause in sequence distinctly from the others.) We varied the number of pauses after each real token from one to three for both CYB and TBYS. Evaluation of all comparison models and CYB used the same subset of 160k sequences from a validation set of C4.

Baselines. We tested a *baseline* model with no pause tokens inserted in the sequences. The baseline model thus had the same number of real tokens per input sequence but $1/M$ the overall length of the CYB and TBYS models with $M - 1$ pauses. Thus, the baseline is data matched but lower in memory and compute demands. We included a second baseline, which we refer to as *baseline 4×seq*, which is memory and compute matched but its context window is filled with $M = 4$ times as many training tokens as CYB and TBYS.

Additional fine-tuning data regimes. Besides the original fine-tuning regime, we ran a subset of methods (baselines and CYB) using $2\times$ and $3\times$ training data as well as using 2 epochs on the original dataset.

D.3 Where and how many pauses are permitted?

Two distinct recipes can be used for training with the CYB losses. We use the first of these, but future research could explore the second. Both require that we specify the maximum number of steps allowed, M.

D.3.1 Recipe 1: Adaptive readout (fixed pause allocation)

The first recipe, which we use in the experiments we report in the main paper, inserts $M-1$ pauses in the input stream after each real input token during training. The transformer steps are run in parallel with causal masking. The pause steps may or may not be used by the model, e.g., if at some step i the model is certain ($d_i \to 0$) or the world enforces a stop ($\widetilde{\omega}_i \to 1$), the outputs at subsequent pauses will not contribute to the loss.

During inference, when the transformer might be run autoregressively, we match the use of pauses at train time, i.e., always inserting $M-1$ pauses after each real input token. Because an input token and the pauses that follow can be processed in parallel, there is no computational cost to inserting the maximum number of pauses in the stream for every token. Rather, the potential cost comes from the pauses cluttering the context window. Even when $M-1$ pauses are inserted at both training and inference, and even when the model is not pressured to reply quickly (e.g., $\gamma_i = 1$ for all i), the model must still learn to use the <DON'T KNOW> output to calibrate itself and determine when to read out.

D.3.2 Recipe 2: Adaptive computation time (dynamic pause insertion)

The second recipe involves a variable number of pause steps at both training and inference. The CYB losses can accommodate variability in the number of pauses following a token at training. No modification to the loss needs to be made, but the <DON'T KNOW> probability must be forced to zero on the last pause, or in response to an input token itself when there are no pauses. Rather than computing the expectation over world stopping times, we can sample a stopping time for each token and insert that number of pause steps in the input stream. One might also envision training methods that insert more pauses where they are likely to be most useful, e.g., where thinking might be required. (In the results section, we present evidence that pause requests by the model are token- and context-specific.)

During inference, the transformer might be run autoregressively, and we can again sample from the world stop-time distribution to determine the number of pause steps, and sample from the model's <DON'T KNOW> confidence to determine which of the allowed steps to read out from.

E Calibration of model confidence and accuracy

Hypothesis **H2** states that CYB should train models to self-calibrate: <DON'T KNOW> should have high probability when a delay in responding will improve model output. Even without discounting ($\gamma_i = 1$) or forced stops ($\boldsymbol{\omega} = (0, \ldots, 0, 1)$), minimizing the CYB loss implies the model should not produce a <DON'T KNOW> response at step i unless accuracy improves at step $i+1$. The next two sections argue for the theoretical justification and empirical evidence supporting this conjecture.

E.1 Theoretical justification

Under $\boldsymbol{\omega} = (0, \ldots, 0, 1)$, Equation 1 simplifies to

$$s_i^k = (1 - d_i^k) \prod_{j=1}^{i-1} d_j^k \text{ with } d_M^k \equiv 0.$$

Defining the expected discounted accuracy recursively as

$$V_i^k = (1 - d_i^k)\gamma_i t_i^k + d_i^k V_{i+1}^k \text{ with } V_M^k = \gamma_M t_M^k \text{ and } V_1^k = \mathbb{E}_{i \sim \Pr(S|\boldsymbol{d}^k, \boldsymbol{\omega})}[\gamma_i t_i^k],$$

the partial derivative of $\ell_{\text{CYB}} = -\log V_1^k$ with respect to the <DON'T KNOW> output is

$$\frac{\partial \ell_{\text{CYB}}}{\partial d_i^k} = -\frac{\prod_{j=1}^{i-1} d_j^k}{V_1^k} \left(V_{i+1}^k - \gamma_i t_i^k\right)$$

Critically, the sign of the derivative depends not on accuracy alone, but on whether the marginal future gain (V_{i+1}^k) is larger than the current discounted accuracy ($\gamma_i t_i^k$). Increasing the <DON'T KNOW> probability will reduce the CYB loss when abstention will lead to a better outcome on expectation. Note that $\gamma_i < 1$ will encourage abstention only when absolute accuracy is increasing. Without discounting, d_i^k will be driven low only for tokens where subsequent pause steps actively degrade expected target accuracy.

E.2 Empirical evidence

To be successful, the model must implicitly predict its own accuracy at subsequent steps. That is, the probability of responding <DON'T KNOW> at step i, d_i, should be large when the target probability is higher at the next step, i.e., $t_{i+1} > t_i$.

Figure E.5 shows joint density maps of <DON'T KNOW> probability (along abscissa) and the improvement in accuracy obtained by a one-step delay in responding (along ordinate). The three density maps correspond to steps $i \in \{1, 2, 3\}$. A positive correlation indicates that the model can predict when it will benefit by the delay. Over the evaluation set, the Spearman correlation coefficient between d_i and $t_{i+1} - t_i$ is 0.383, 0.157, and 0.271 for steps 1-3, respectively, all reliably nonzero ($p < 0.001$).[2] The correlations are only slightly stronger with the discounting variant of CYB ($\gamma = 0.99^{\{0,1,2,3\}}$)—0.381, 0.208, and 0.309—indicating that penalizing delayed responses does not appear to make the model much better at determining when to pause.

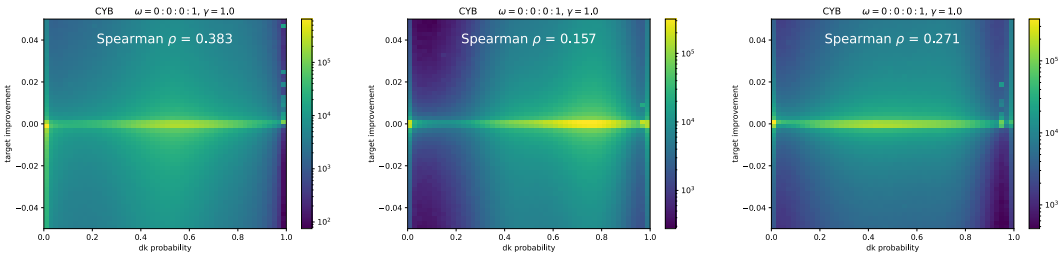

Figure E.5: Density plots indicating that <DON'T KNOW> probability is positively correlated with improvement in accuracy, for steps 1, 2, and 3.

[2]The statistical reliability of the non-zero correlation coefficient is partly due to the large sample size used to compute the correlation.

F Additional results

In this section, we present additional results.

F.1 Tabular representations of the results from the main paper

In this section, we present the results from Figure 3 and Figure 4 in a tabular format for the easy introspection. We report C4 eval set perplexity as well as the relative perplexity improvement (in %) compared to the baseline perplexity. Relative improvement is computed as the percentage decrease in perplexity relative to the TBYS with 3 pause tokens, i.e.

$$\text{Relative Improvement} = (\text{PPL}_{\text{TBYS,p=3}} - \text{PPL}_{\text{method}})/\text{PPL}_{\text{TBYS,p=3}} \times 100\% \tag{5}$$

The results for fine-tuning of Gemma3-4B model are given in Table F.2. We see that even CYB with 1 pause outperforms TBYS with 3 pauses. Using 3 pauses with CYB improves performance even further leading to **+2.2%** relative improvement in perplexity. The results for fine-tuning of Gemma1-2B model are given in Table F.3. Overall, the behavior is very similar to Gemma3-4B experiment. Finally, the results of pretraining of Gemma1-2B model are given in Table F.4. What is remarkable is that using CYB with the pretrained model leads to even higher relative improvement in perplexity, notably **+3.57%** compared to the fine-tuned model which enjoys **+2.72%** relative improvement.

Method	**Perplexity**	**Rel. Improvement (%)**
Pretrained Gemma3-4B	11.765	-13.57%
Baseline	10.670	-3.00%
Baseline 4×seq	10.694	-3.24%
TBYS 1 pause	10.512	-1.48%
TBYS 2 pauses	10.422	-0.61%
TBYS 3 pauses	10.359	—
CYB 1 pause	10.351	+0.08%
CYB 2 pauses	10.210	+1.44%
CYB 3 pauses	**10.131**	**+2.20%**

Table F.2: **Results for fine-tuned (on C4) Gemma3-4B**. Perplexity on eval set of C4. For relative improvement, we use the formula (5). In **bold** we highlight the best result – the lowest perplexity and the highest relative improvement.

Method	**Perplexity**	**Rel. Improvement (%)**
Pretrained Gemma1-2B	13.018	-8.64%
Baseline	12.173	-1.58%
Baseline 4×seq	11.953	+0.25%
TBYS 1 pause	12.098	-0.96%
TBYS 2 pauses	12.024	-0.34%
TBYS 3 pauses	11.983	—
CYB 1 pause	11.865	+0.99%
CYB 2 pauses	11.758	+1.88%
CYB 3 pauses	**11.657**	**+2.72%**

Table F.3: **Results for fine-tuned (on C4) Gemma1-2B**. Perplexity on eval set of C4. For relative improvement, we use the formula (5). In **bold** we highlight the best result – the lowest perplexity and the highest relative improvement.

Method	Perplexity	Rel. Improvement (%)
Pretrained Gemma1-2B	13.018	-4.14%
Baseline	12.893	-3.14%
TBYS 3 pauses	12.500	—
CYB 3 pauses	**12.054**	**+3.57%**

Table F.4: **Results for trained from scratch (on C4) Gemma1-2B**. Perplexity on eval set of C4. For relative improvement, we use the formula (5). In **bold** we highlight the best result – the lowest perplexity and the highest relative improvement.



Figure F.6: Model trained with CYB loss produces variable pause durations on tokens.

G Derivation of output probability

Based on Figure 2, the probability of outputting a result at step i is given by

$$\Pr(S = i|\boldsymbol{d}, \boldsymbol{\omega}) = \left(\prod_{j=1}^{i-1} d_j\right)\left(\prod_{j=1}^{i-1}(1-\widetilde{\omega}_j)\right)(d_i\widetilde{\omega}_i + (1-d_i)) \tag{6}$$

Using the definition of the survival probability in Figure 2,

$$\tilde{\omega}_i = \omega_i / \textstyle\sum_{j=i}^{M} \omega_j \tag{7}$$

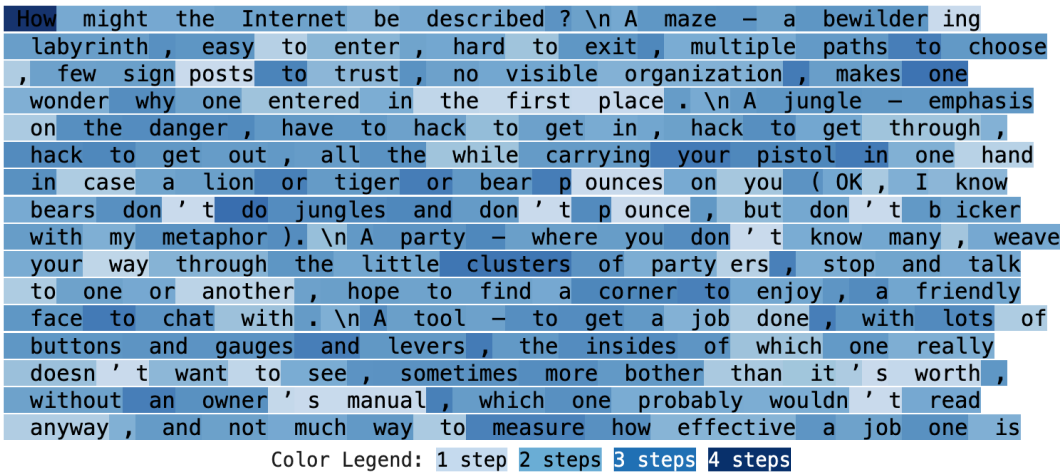

Figure F.7: Model trained with CYB loss produces variable pause durations on tokens.

we notice that

$$\begin{aligned} d_i\widetilde{\omega}_i + (1-d_i) &= \frac{1}{\sum_{k=i}^{M}\omega_k}\left(d_i\omega_i + (1-d_i)\sum_{k=i}^{M}\omega_k\right) \\ &= \frac{1}{\sum_{k=i}^{M}\omega_k}\left(d_i\omega_i + \omega_i - d_i\omega_i + (1-d_i)\sum_{k=i+1}^{M}\omega_k\right) \\ &= \frac{1}{\sum_{k=i}^{M}\omega_k}\left(\omega_i + (1-d_i)\sum_{k=i+1}^{M}\omega_k\right) \end{aligned}$$

Moreover, we also notice that

$$\begin{aligned} \prod_{j=1}^{i-1}(1-\widetilde{\omega}_j) &= \prod_{j=1}^{i-1}\frac{\sum_{k=j+1}^{M}\omega_k}{\sum_{k=j}^{M}\omega_k} && (8) \\ &= \frac{\prod_{j=1}^{i-1}\sum_{k=j+1}^{M}\omega_k}{\prod_{j=1}^{i-1}\sum_{k=j}^{M}\omega_k} && (9) \\ &= \left(\sum_{k=i}^{M}\omega_k\right)\left(\frac{1}{\sum_{k=1}^{M}\omega_k}\right)\left(\frac{\prod_{j=2}^{i-1}\sum_{k=j}^{M}\omega_k}{\prod_{j=2}^{i-1}\sum_{k=j}^{M}\omega_k}\right) && (10) \\ &= \sum_{k=i}^{M}\omega_k, && (11) \end{aligned}$$

where we used the fact that $\sum_{k=1}^{M}\omega_k = 1$. Now, we plug these two into (6) and we get

$$\Pr(S=i|\boldsymbol{d},\boldsymbol{\omega}) = \left(\prod_{j=1}^{i-1}d_j\right)\left(\omega_i + (1-d_i)\sum_{k=i+1}^{M}\omega_k\right) \quad (12)$$

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Paper focus is on showing that our proposed loss, CYB, improves performance over the current practice for models with pause tokens. Plenty of evidence in support.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See discussion and future work section.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: [N/A]

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Main paper gives the big picture description; supplementary material provides the details, including learning rates, amount of training, etc.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: While we do not include the code implementation, we include extensive experimental details required to reproduce the results from the paper. We anticipate—with employer permission—to release the code and weights on publication.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: To the best of our knowledge, we have provided all the details of training and evaluation needed to understand and replicate our results. Some of the details needed to replicate appear in the supplementary materials, but we did our best to ensure relevant details of training needed to understand the work are in the main text.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report replications in several sections involving fine tuning and downstream tasks. We did not have the compute capability to run replications everywhere. However, where we have replications, we go further than producing error bars and perform paired t-tests to evaluate reliability. We also perform two-tailed Student's t-tests for determining the reliability of a nonzero correlation. We could calculate error bars for evaluations by treating individual problems as the random factor and applying McNemar's Test (see Williams, 2026).

However, these evaluation sets are large enough that any small reliable difference will be detected.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide all the experimental details in Section D and Section 4, which includes the hyperparameters, number of iterations, architecture and the used hardware. Caveat: We do not provide execution times.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We fulfill all requirements.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact above and beyond the generic concerns about the effect of AI on society. The proposed loss is a minor change on traditional losses used for LLMs and is intended to improve computation through the use of pause tokens.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our work focuses on a new loss for pretraining or fine tuning and therefore presents no particular issues of misuse.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We credit the creators of each dataset used in our experiments.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new data assets. Primary contribution is an algorithm which is precisely specified in the main text.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We do not use human data.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: We do not conduct experiments on human participants.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: We have not made use of LLMs in any significant way beyond help editing individual sentences.